МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ, ЧИСЛЕННЫЕ МЕТОДЫ И КОМПЛЕКСЫ ПРОГРАММ

О ВЛИЯНИИ КЭШ-ПАМЯТИ НА ЭФФЕКТИВНОСТЬ ПРОГРАММНОЙ РЕАЛИЗАЦИИ БАЗОВЫХ ОПЕРАЦИЙ ЛИНЕЙНОЙ АЛГЕБРЫ[1]

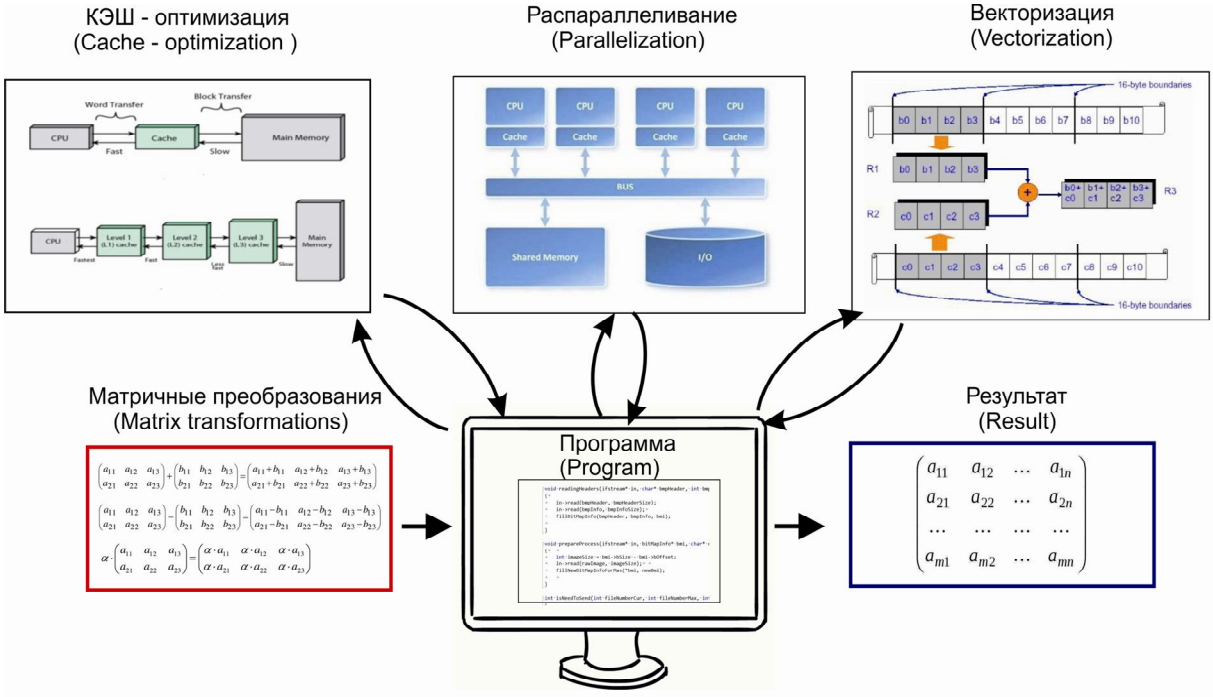
Статья поступила в редакцию 06.12.2018, в окончательном варианте – 26.12.2018.

Егунов Виталий Алексеевич, Волгоградский государственный технический университет, 400005, Российская Федерация, г. Волгоград, пр.им. Ленина, 28,
кандидат технических наук, доцент, e-mail: vegunov@mail.ru

Рассматриваются вопросы повышения эффективности программной реализации базовых операций линейной алгебры на параллельных вычислительных системах с общей памятью. Выбор подобных систем обусловлен их широкой распространенностью и доступностью в настоящее время. Для согласования скоростей работы устройств в составе вычислительной системы применяется иерархическая структура памяти, в состав которой кроме основной памяти входит многоуровневая кэш-память. Организация эффективного взаимодействия микропроцессора с такой кэш-памятью оказывает огромное влияние на итоговое время работы программы. В статье рассматриваются вопросы оптимизации доступа к данным при выполнении базовых операций линейной алгебры над большими массивами данных. Показано, что подобная оптимизация может существенно сократить итоговое время выполнения подобных преобразований. Приводится методика аналитической оценки эффективности подобных программ. Сделанные выводы подтверждаются результатами проведенных вычислительных экспериментов.

Ключевые слова: многоядерные процессоры, многопроцессорные системы, неоднородные параллельные вычислительные системы, эффективность программ, ускорение работы программ, кэш-память, системы с общей памятью, линейная алгебра, операции линейной алгебры, векторные операции

Графическая аннотация (Graphical annotation)



[1] Исследование выполнено при финансовой поддержке РФФИ и Администрации Волгоградской области в рамках научного проекта № 18-47-340010 р_а.

ON THE INFLUENCE OF CACHE MEMORY ON THE EFFECTIVENESS OF PROGRAMME IMPLEMENTATION OF THE BASIC LINEAR ALGEBRA OPERATIONS

The article was received by editorial board on 06.12.2018, in the final version – 26.12.2018.

Egunov Vitaly A., Volgograd State Technical University, 28 Lenin Ave., Volgograd, 400005, Russian Federation,

Cand. Sci. (Engineering), e-mail: vegunov@mail.ru

The problems of increasing the efficiency of software implementation of basic operations of linear algebra on parallel computing systems with shared memory are considered. The choice of such systems is due to their widespread availability at the present time. To match the speed of the devices in the computing system uses a hierarchical memory structure, which in addition to its own main memory is a multi-level cache memory. The organization of effective interaction of the microprocessor with the cache memory has a huge impact on the final execution time of the program. The paper deals with the optimization of data access when performing basic linear algebra operations on large data sets. It is shown that such optimization can significantly reduce the total conversion time. The method of analytical evaluation of the effectiveness of such programs is presented, the findings are confirmed by the results of computational experiments.

Key words: multi-core processors, multiprocessor systems, heterogeneous parallel computing systems, program efficiency, program acceleration, cache memory, shared memory systems, linear algebra, linear algebra operations, vector operations

Введение. Эффективная программная реализация базовых операций линейной алгебры для вычислительных систем всегда считалась актуальной задачей. Практически во всех системах моделирования в различных областях матричные операции являются их основой. Задачи линейной алгебры играют огромную роль в современных научных исследованиях и при решении большого числа прикладных проблем. Часто именно на матричные преобразования, такие как решение систем линейных алгебраических уравнений, поиск собственных значений и системы собственных векторов, поиск сингулярного разложения, обращение матриц и т.д., приходится основная часть операционной сложности решения. Публикуется большое количество работ, посвященных решению конкретных задач линейной алгебры на различных параллельных архитектурах, в том числе неоднородных. Так, работы [9, 18, 21] посвящены исследованию проблемы собственных значений матрицы, в работе [8] рассматривается QR – разложение, в работах [11, 12] LU – разложение исходных матриц. Работы [13, 15] посвящены обработке разреженных матриц и другим матричным преобразованиям [6, 7, 10, 16, 17, 19, 20]. Рассматриваются вопросы решения данных задач на различных параллельных архитектурах, в т.ч. на неоднородных многопроцессорных вычислительных системах.

При решении всех приведенных выше задач используются простейшие базовые операции линейной алгебры, такие как вычисление скалярного произведения векторов, поэлементное сложение векторов, нахождение поэлементного среднего нескольких векторов, умножение матриц и т.д. Эффективность реализации данных операций оказывает серьезное влияние на производительность программ, выполняющих более сложные операции линейной алгебры, таких как, например, решение систем линейных алгебраических уравнений (СЛАУ), поиск собственного разложения и т.д.

Данная статья написана во многом по следам VII Молодежной школы по робототехнике, искусственному интеллекту и инженерному творчеству «Робошкола++», проведенной на кафедре ЭВМ и систем Волгоградского государственного технического университета в период с 19 по 24 ноября 2018 г. Участники делились по нескольким «трекам», в рамках которых прослушали лекции преподавателей школы, поучаствовали в практических занятиях, а в завершающей стадии – в хакатоне, где они соревновались в решении предложенной организаторами задачи. В рамках трека «Высокопроизводительные вычисления» участникам была предложена задача обработки массива изображений, при решении которой необходимо было продемонстрировать навыки использования различных технологий распараллеливания и векторизации вычислений. Одной из основных простейших операций в данном случае являлась операция нахождения поэлементного среднего для значений нескольких пикселей исходного изображения.

Общая характеристика проблематики работы. Одной из задач хакатона являлась разработка программы, выполняющей масштабирование изображения, при решении которой участники усредняли значения для каналов R, G, B исходного изображения. Данную задачу можно охарактеризовать как задачу разработки параллельной программы для системы с общей памятью. Системы с общей памятью привлекают большое число исследователей из-за своей широкой распространенности и доступности в настоящее время. Проблеме разработки эффективных программ для подобных систем посвящено большое число исследований, результаты которых отражены в различных публикациях, например в [1, 2, 5] и ряде других.

Рассмотрим данную задачу подробнее. Сам алгоритм решения задачи вопросов не вызывал. Основную сложность представлял вопрос организации порядка обработки исходных пикселей, так как здесь возможны несколько вариантов. Они представлены на рисунке 1.

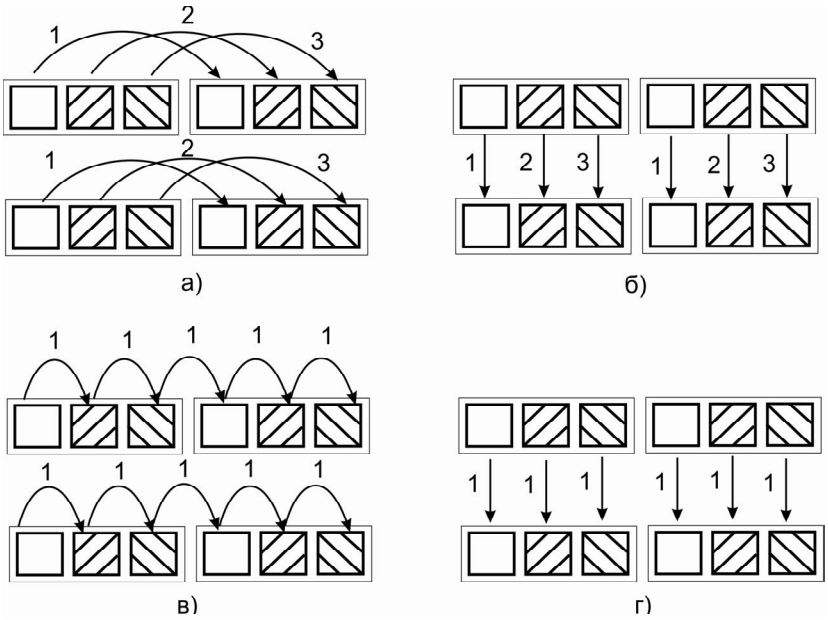


Рисунок 1 – Варианты порядка обхода пикселей в массиве изображений

Возможны четыре варианта обхода пикселей исходного изображения. При этом выбор любого из них не влияет на конечный результат, однако влияет на время выполнения преобразования.

Две вычислительные схемы реализации данного алгоритма предполагают последовательную обработку цветовых каналов. Т.е. сначала формируются все биты R нового изображения, затем все биты G, затем B. На рисунке 1а представлен такой подход при проходе изображения по строкам, на рисунке 1б – по столбцам. Таким образом, для формирования результирующего изображения необходимо три прохода по исходному массиву изображения. Необходимо отметить, что сами изображения в файле формата *.BMP, обработка которых была предложена в качестве задания к хакатону, хранятся по строкам. Поэтому сначала в файл записываются все значения всех пикселей одной строки изображения, затем второй, и т.д. Сами пиксели представлены в виде трех байтов, хранящих изображения для цветовых каналов R, G, B (на рис. 1 для изображения различных цветовых каналов используются различные варианты штриховки).

Две другие вычислительные схемы выполняются за один проход исходного массива. Байты цветовых каналов пикселей обрабатываются последовательно, накопление результата осуществляется в разных переменных для отдельных цветовых каналов. При этом вариант 1в представляет собой обработку исходного массива по строкам, 1г – по столбцам.

Все четыре вычислительных схемы требуют одинакового числа операций обращения к памяти, имеют равную операционную сложность. Однако программы, построенные на основе различных схем, имеют различную вычислительную эффективность. Другими словами, время их выполнения при обработке одного и того же массива изображений отличается.

Методика оценки эффективности программ, выполняемых на вычислительных системах с общей памятью. Оценим эффективность рассматриваемых программ с использованием методики, предложенной в работе [4]. За основу аналитической оценки времени выполнения программы была взята методика, предложенная в [3]. Время выполнения программы, согласно данной методике, можно оценить следующим образом (1):

$$T = nOp \cdot t + nMem \cdot g \cdot (a + {}^{64}/_{b}), \quad (1)$$

где nOp – число операций, необходимых для реализации алгоритма; t – среднее время выполнения одной операции; nMem – объем извлекаемых из памяти данных; g – частота кэш – промахов; a – латентность оперативной памяти; b – пропускная способность оперативной памяти.

Константа «64» определяет ширину строки кэш-памяти в байтах. В данной работе уделим внимание проблеме снижения частоты кэш-промахов. Кэш-промах происходит в том случае, когда микропроцессор пытается обратиться к данным, которые отсутствуют в кэш-памяти. В этом случае их необходимо подгружать из основной памяти. Если необходимые данные находятся в кэш-памяти, они быстро извлекаются. Такое событие называется кэш-попаданием. Повышение быстродействия вычислительной системы достигается в том случае, когда кэш-попадания реализуются намного чаще, чем кэш-промахи.

В приведенных выше алгоритмах частоту кэш-промахов можно оценить как вероятность отсутствия требуемых данных в кэш-памяти при выполнении очередной итерации алгоритма усреднения значений исходных матриц. Для получения достоверных результатов, вычисления повторялись многократ-

но, данные усреднялись. Поэтому, в том случае, когда матрицы исходного и результирующего изображений «должны» помещаться в кэш, их загрузкой на первом проходе можно пренебречь. В случае, если суммарный объем данных не превышает объема кэш-памяти, то после первого прохода данные целиком размещены в кэш-памяти, и для осуществления второго и последующих проходов к общей памяти обращаться нет необходимости. Если же массивы целиком в кэш-память не помещаются, то наблюдается иная картина. К началу второго прохода определенное количество элементов массивов уже вытеснены из кэш-памяти, их необходимо загружать заново.

Данные загружаются в кэш-память строками по 64 байта. Таким образом, частота кэш-промахов будет зависеть от базового типа данных. Например, при работе с данными типа double эта частота будет равна 1 / 8. При работе с однобайтными типами данных, например char, она будет равна 1 / 64, хотя в реальности она будет несколько выше. Необходимо отметить, что выбор схемы 1а или 1б в используемом алгоритме может оказать влияние на частоту кэш-промахов только в случае выбора массивов, которые имеют размер по «вертикали» достаточный, чтобы столбцы исходных массивов полностью заняли весь объем кэш-памяти. Тогда к тому моменту, когда обработан последний элемент в первом столбце, строка кэш – памяти, содержащая его первый элемент будет уже вытеснена в основную память и ее придется подгружать заново. В этом случае частота кэш-промахов для результирующего массива может вырасти до 1, что приведет к необходимости обращаться к основной памяти каждый раз, когда необходима загрузка нового элемента. Для исходного массива частота кэш-промахов всегда будет меньше 1. Причина: при усреднении всегда обрабатывается какое-то количество элементов, находящихся рядом, т.е. в одних и тех же строках кэш-памяти.

На практике, однако, ситуация будет несколько иной, частота кэш-промахов может вырасти и при меньшем размере массива «по вертикали». Это связано с тем, что в современных процессорах не используется полностью ассоциативный кэш, где и наблюдалась бы такая картина. На практике применяется множественно-ассоциативный кэш, число каналов которого для кэша L3 варьируется от 4 до 16. Это означает, что если считываются данные, которые не расположены в памяти последовательно друг за другом, то заполнение одного из каналов может произойти раньше заполнения всего кэша целиком.

Таким образом, в схеме 1б частота промахов будет варьироваться от 1 / 8 до 1, среднее значение данного параметра будет находиться между этими значениями. Потенциально схема 1б является наименее эффективной.

Вычислительные схемы 1в и 1г являются однопроходными. Поэтому характеристики программ, основанных на них, определить достаточно легко, опираясь на рассуждения, приведенные выше.

Описание задачи. Несколько модифицируем исходную задачу. Во-первых, с целью ее обобщения, чтобы в дальнейшем не привязываться к конкретной задаче масштабирования изображений. Во-вторых, чтобы несколько увеличить операционную сложность алгоритма.

За основу возьмем подход, используемый в методе конечных разностей, одном из сеточных методов решения дифференциальных уравнений. Его основная идея заключается в дискретизации первоначально непрерывных значений координаты и времени, с помощью которой дифференциальное уравнение может быть сведено к системе алгебраических, решение которых оказывается значительно проще дифференциального. В данной работе отойдем от собственно решения дифференциального уравнения, будем использовать методику, при которой расчетная область покрывается сеткой точек.

В качестве базовой задачи возьмем обработку массива данных, полученных в результате дискретизации значений некоей скалярной функции двух переменных, которая для простоты имеет прямоугольную область определения. В качестве примера можно привести распределение температуры по поверхности плоского прямоугольного листа. Общий объем информации в данном случае будет зависеть от шага дискретизации, т.е. от размера сетки. Над данным массивом данных можно осуществлять различные вычисления, например пересчет температур со временем, в связи с изменением граничных условий и т.д. Одной из задач может являться то же масштабирование для изменения размера сетки и ряд других задач. Предположим теперь, что с каждой точкой плоскости связано несколько значений, которые представляют собой различные характеристики. Каждая характеристика будет представлять собой аналог цветового канала в задаче обработки изображений. Сами характеристики будут представлены значениями типа double. Рассмотрим задачу усреднения значений подобных систем по всем каналам с использованием вычислительных схем, описанных выше и представленных на рисунке 1. По сути, основными операциями, необходимыми в данном случае, являются простейшие базовые операции линейной алгебры, такие как сложение векторов, масштабирование элементов векторов.

Для того чтобы оценить значение выражения (1), необходимо определиться со значением ключевых параметров – числа операций, частоты кэш-промахов, числа операций. Общий объем загружаемых данных во всех случаях одинаков и зависит от размера массивов, числа каналов, размера усредняемой области:

$$nMem = width \cdot height \cdot N \cdot L \cdot (1 + k \cdot k), \quad (2)$$

где width, height – размеры массива, определяемые размерами области определения исходных функций, формирующих обрабатываемые каналы в массиве данных; N – число обрабатываемых каналов; L – длина базового типа в байтах (для типа double равна 8); k – размер усредняемой области.

Для того чтобы использовать выражение (1), необходимо определиться с числом операций nOp. Очевидно, что операционная сложность не зависит от порядка обработки данных. Если использовать задачу усреднения данных по всем каналам, то операционную сложность можно определить следующим образом:

$$nOp = width \cdot height \cdot N \cdot (k \cdot k + addOp), \quad (3)$$

где k – размер локальной области, на которой осуществляется усреднение, например $3 \cdot 3$, $5 \cdot 5$ и т.д.; addOp – число дополнительных операций, которые необходимо выполнить для вычисления каждого усредненного значения (оно зависит от кода программы).

Результаты вычислительных экспериментов. В качестве объекта исследований была выбрана параллельная вычислительная система с общей памятью, построенная на базе многоядерного микропроцессора Intel Core i7 – 2600 K, 3500 MHz. Данный микропроцессор построен по технологии Sandy Bridge, содержит 4 ядра, поддерживается технология Hyper – Threading, поддерживается двухканальная память DDR3 – 1333. Микропроцессор имеет L3-кэш объемом 8 Мб. При этом используется технология Smart Cache, что предполагает динамическое эффективное распределение кэш-памяти между ядрами процессора. Более подробно о специфике оценки эффективности параллельных программ на конкретной вычислительной системе можно прочитать в [4].

В таблице 1 приведены результаты, полученные для программы, осуществляющей поэлементное преобразование содержимого одного массива в элементы другого массива. По сути, данная программа реализует алгоритм усреднения по четырем каналам при размере усредняемой области размером в один элемент.

Таблица 1 – Время работы программы, полученное в результате вычислительных экспериментов

Размеры массивов	Объем данных, Мб	Время вычислений, с (4 канала, размер области 1 · 1)			
		1а	1б	1в	1г
50 · 50	0.16	0,03	0,03	0,03	0,03
74 · 74	0,35	0,08	0,082	0,079	0,08
100 · 100	0.64	0,13	0,15	0,13	0,13
124 · 124	0,98	0,3	0,31	0,28	0,29
150 · 150	1,44	0,45	0,48	0,43	0,44
174 · 174	1,94	0,62	0,66	0,6	0,61
200 · 200	2,56	0,8	1,9	0,5	1,9
320 · 320	6,55	3	8	1,5	7
496 · 496	15,7	8,5	29	4,5	22
1000 · 1000	64	32	114	16	88
1500 · 1500	144	74	377	57	328

Массивы, обрабатываемые в программе, выравнивались по адресу 64. Размерность массивов подбиралась таким образом, чтобы все строки массива располагались с адресов, кратных 64, т.е. совпадали с началом строки, подгружаемой в кэш-память. Анализ данных в этой таблице подтверждает выводы, сделанные выше на основе методики, основанной на аналитической оценке времени выполнения программы:

- пока массивы целиком помещаются в кэш-память L3, время выполнения программы практически не зависит от алгоритма обработки (первые 6 строк таблицы);
- более эффективными оказываются алгоритмы 1в и 1г;
- в каждой паре более эффективным оказывается алгоритм, обрабатывающий массивы по строкам.

Некоторая разница по времени обработки массивов, целиком помещающихся в кэш-память L3, обусловлена наличием кэшей L2 и L1, которые имеют меньший размер. При взаимодействии между уровнями кэшей L2 и L3 наблюдаются те же эффекты, что и при взаимодействии кэш-памяти L3 и основной памяти. Поэтому время выполнения программ несколько отличается. Стоит отметить, что значения в первой строке одинаковы из-за того, что в данном случае все данные помещаются уже в кэш L2, который для данного микропроцессора имеет объем 256 K. Для наглядности результаты, представленные в таблице 1, приведены на рисунке 1.

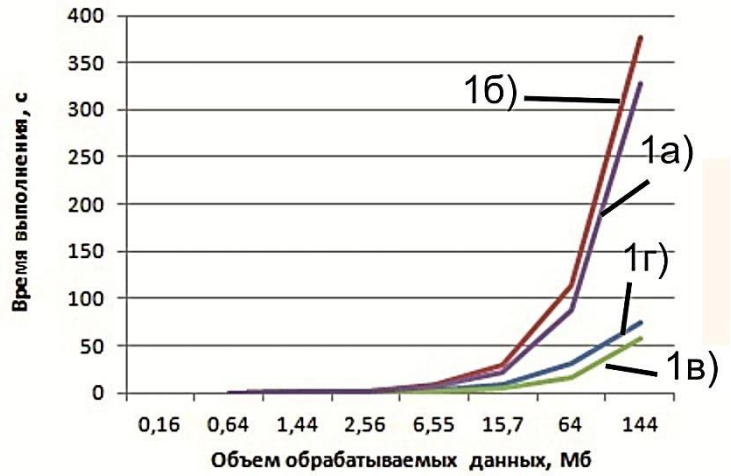


Рисунок 1 – Время выполнения программы при усреднении по четырем каналам, область усреднения 1 · 1

Если изменить параметры алгоритма, то картина изменится. На рисунке 2 представлено время выполнения программы при усреднении на области 5 · 5 по четырем каналам.

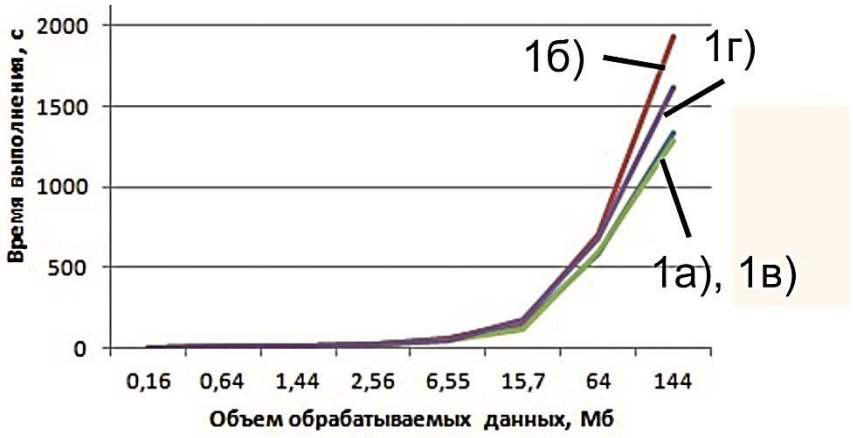


Рисунок 2 – Время выполнения программы при усреднении по четырем каналам, область усреднения 5 · 5

Видно, что лучшими алгоритмами в данном случае являются строчные схемы 1а и 1в (в масштабе, использованном на рисунке 2, они сливаются, хотя лучшим является алгоритм 1в). «Столбцовые» схемы им проигрывают, хотя разница во времени выполнения не так велика, как на рисунке 1. Алгоритмы, результаты для которых отражены на рисунке 2, гораздо ближе друг к другу по эффективности. Это происходит из-за того, что при усреднении существенно падает частота кэш-промахов для исходного массива. Причина: осуществляется считывание прямоугольной области, которая «перемещается» по массиву. Соответственно необходимые данные чаще оказываются в кэш-памяти. С увеличением области усреднения будет происходить дальнейшее выравнивание вычислительной эффективности программ.

На рисунке 3 представлено время выполнения программы при усреднении на области 2 · 2 по 32 каналам. Параметры были сознательно выбраны таким образом, чтобы соседние элементы (с учетом каналов) гарантировано находились в различных строках кэш-памяти. Видно, что в данном случае столбцовый алгоритм 1г является более эффективным, чем строчный алгоритм 1а.

В большом количестве различных публикаций можно встретить утверждение, что данные необходимо пытаться в любом случае обрабатывать в том порядке, в каком они хранятся в памяти. Обрабатываемые массивы хранятся по строкам, однако, столбцовый алгоритм 1г на рисунке 2 оказался эффективнее строчного алгоритма 1а.

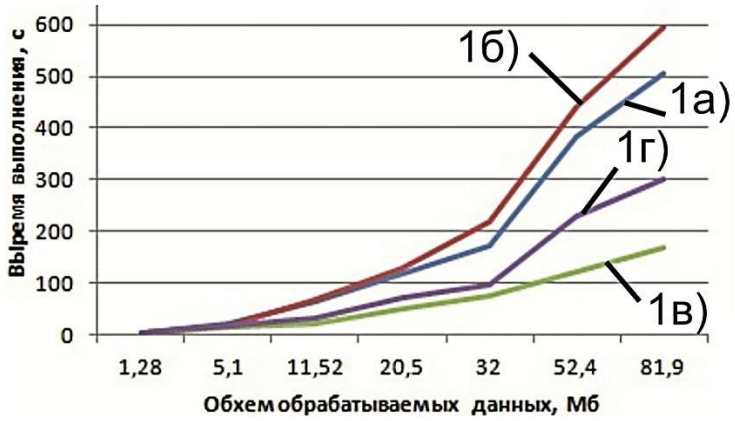


Рисунок 3 – Время выполнения программы при усреднении по 32-м каналам, область усреднения 2 · 2

Выводы. 1. При разработке программ необходимо учитывать, что большое влияние на время их выполнения оказывает взаимодействие микропроцессора с кэш-памятью, являющейся обязательной частью всех современных микропроцессоров. Критически важной характеристикой в этом случае необходимо считать частоту кэш-промахов, при которых необходимые данные отсутствуют в быстрой кэш-памяти. Поэтому возникает необходимость обращения к относительно «медленной» основной памяти. **2**. Для повышения вычислительной эффективности работы программ, в т.ч. программ реализации базовых операций линейной алгебры, необходимо осуществить анализ выбранного алгоритма доступа к данным. Такой анализ должен предшествовать собственно разработке программы. **3**. Наиболее эффективный алгоритм доступа к данным может отличаться от способа хранения данных, т.е. при построчном хранении данных при определенных условиях более эффективным может оказаться алгоритм постолбцовой обработки, как это было показано выше. **4**. Алгоритм доступа к данным должен впоследствии хорошо распараллеливаться и подвергаться векторизации. **5**. В ряде случаев может быть целесообразным изменить базовый алгоритм решения задачи для повышения вычислительной эффективности полученной программы, как это показано, например, в [14] на примере реализации QR и LQ разложений.

Библиографический список

1. Андреев А.Е., Егунов В.А., Шаповалов О.В. Технологии программирования многопроцессорных систем. / Учебное пособие / Волгоград, 2015.

2. Галанин М.П., Лукин В.В., Чечеткин В.М. Моделирование астрофизических струйных выбросов на гибридных вычислительных системах с общей памятью. / В сборнике параллельные вычислительные технологии (ПаВТ2012) Труды международной научной конференции. Ответственные за выпуск Л.Б.Соколинский, К.С. Пан. 2012. С.110.

3. Гергель В.П. Высокопроизводительные вычисления для многопроцессорных многоядерных систем: Учебник – М.: Издательство Московского университета, 2010. – 544 с., илл. – (Серия «Суперкомпьютерное образование»).

4. Егунов В.А. Оценка эффективности программной реализации QR-разложения на многоядерных архитектурах / В.А. Егунов, С.И. Кирносенко, П.Д. Кравченя, О.О. Шумейко // Известия ВолгГТУ. Сер. Актуальные проблемы управления, вычислительной техники и информатики в технических системах. - Волгоград, 2017. - № 1 (196). - С. 56-59.

5. Иванов А.М., Хохлов Н.И. Применение технологий параллельного программировании для систем с общей памятью при решении гиперболических систем уравнений. / Труды Московского физико-технического института. 2016. Т.8 №2 (30). С.101-111.

6. Ильин В.П. Проблемы высокопроизводительных технологий решений больших разреженных СЛАУ. //Вычислительные методы и программирование: новые вычислительные технологии. 2009. Т.10 №1. С.141-147.

7. Кочура А. Е., Подкользина Л. В., Ивакин Я. А., Нидзиев И. И. Разработка алгоритма решения систем линейных уравнений с варьируемыми параметрами, использующего разреженность матрицы //Прикаспийский журнал: управление и высокие технологии- 2014-№2

8. Anderson M. Communication-avoiding QR decomposition for GPUs / Anderson, M., Ballard, G., Demmel, J., Keutzer, K.,(2011) Proceedings - 25th IEEE International Parallel and Distributed Processing Symposium, IPDPS 2011, art. no. 6012824, pp. 48-58. ISBN: 978-076954385-7

9. Baker C.G. Anasazi software for the numerical solution of large-scale eigenvalue problems / Baker, C.G., Hetmaniuk, U.L., Lehoucq, R.B., Thornquist, H.K., (2009) ACM Transactions on Mathematical Software, 36 (3), art. no. 13. doi: 10.1145/1527286.1527287

10. Blackford L.S. An Updated Set of Basic Linear Algebra Subprograms / Blackford, L.S., Demmel, J., Dongarra, J., Duff, I., Hammarling, S., Henry, G., Heroux, M., Whaley, R.C., (2002) ACM Transactions on Mathematical Software, 28 (2), pp. 135-151. DOI: 10.1145/567806.567807

11. Chen C. LU factorization on heterogeneous systems: an energy-efficient approach towards high performance / Chen, C., Fang, J., Tang, T., Yang, C. , 2017, Computing, 99(8), pp. 791-811

12. Chow E. Fine-grained parallel incomplete LU factorization / Chow, E., Patel, A., (2015) SIAM Journal on Scientific Computing, 37 (2), pp. C169-C193. doi: 10.1137/140968896

13. Demmel J. Avoiding communication in sparse matrix computations / Demmel, J., Hoemmen, M., Mohiyuddin, M., Yelick, K., (2008) IPDPS Miami 2008 - Proceedings of the 22nd IEEE International Parallel and Distributed Processing Symposium, ISBN: 978-142441694-3 doi: 10.1109/IPDPS.2008.4536305

14. Egunov V.A.. Implementation of QR and LQ decompositions on shared memory parallel computing systems [Электронный ресурс] / V.A. Egunov, A.E. Andreev // 2016 2nd International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM) (Chelyabinsk, Russia, 19-20 May 2016). – [Publisher: IEEE], 2016. – 5 p. – DOI: 10.1109/ICIEAM.2016.7911607.

15. Kreutzer M. A unified sparse matrix data format for efficient general sparse matrix-vector multiplication on modern processors with wide simd units / Kreutzer, M., Hager, G., Wellein, G., Fehske, H., Bishop, A.R., (2014) SIAM Journal on Scientific Computing, 36 (5), pp. C401-C423. doi: 10.1137/130930352

16. Kreutzer M. Building Blocks for High Performance Sparse Linear Algebra on Heterogeneous Systems / Kreutzer, M., Thies, J., Röhrig-Zöllner, M., Shahzad, F. , Pieper, A. Galgon, M., Basermann, A., Fehske, H , Shahzad, F., 2017, International Journal of Parallel Programming 45(5), pp. 1046-1072

17. Kreutzer M. Performance Engineering of the Kernel Polynomal Method on Large-Scale CPU-GPU Systems / Kreutzer, M., Pieper, A., Hager, G., Wellein, G., Alvermann, A., Fehske, H. (2015) Proceedings - 2015 IEEE 29th International Parallel and Distributed Processing Symposium, IPDPS 2015, art. no. 7161530, pp. 417-426. ISBN: 978-147998648-4 doi: 10.1109/IPDPS.2015.76

18. Luciani X. Joint Eigenvalue Decomposition of Non-Defective Matrices Based on the LU Factorization With Application to ICA / Luciani, X., Albera, L., (2015) IEEE Transactions on Signal Processing, 63 (17), art. no. 7118226, pp. 4594-4608. doi: 10.1109/TSP.2015.2440219

19. Monakov A. Automatically tuning sparse matrix-vector multiplication for GPU architectures / Monakov, A., Lokhmotov, A., Avetisyan, A. , (2010) Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 5952 LNCS, pp. 111-125. ISBN: 3642115144; 978-364211514-1 doi: 10.1007/978-3-642-11515-8_10

20. Nelson T. Reliable generation of high-performance matrix algebra / Nelson, T., Belter, G., Siek, J.G., Jessup, E., Norris, B. (2015) ACM Transactions on Mathematical Software, 41 (3), art. no. 18. doi: 10.1145/2629698

21. Polizzi E. Density-matrix-based algorithm for solving eigenvalue problems / (2009) Physical Review B - Condensed Matter and Materials Physics, 79 (11), art. no. 115112. doi: 10.1103/PhysRevB.79.115112

References

1. Andreev A.E., Egunov V.A., Shapovalov O.V. Tekhnologii programmirovaniya mnogoprocessornyh system [Programming technologies for multiprocessor systems]. / Uchebnoe posobie [Tutorial] / Volgograd, 2015.

2. Galanin M.P., Lukin V.V., Chechetkin V.M. Modelirovanie astrofizicheskih strujnyh vybrosov na gibridnyh vychislitel'nyh sistemah s obshchej pamyat'yu [Modeling of astrophysical jet emissions on hybrid computing systems with shared memory]. / V sbornike parallel'nye vychislitel'nye tekhnologii [In collection of parallel computing technologies] (PaVT2012) Trudy mezhdunarodnoj nauchnoj konferencii [Proceedings of the international scientific conference]. 2012. p.110.

3. Gergel' V.P. Vysokoproizvoditel'nye vychisleniya dlya mnogoprocessornyh mnogoyadernyh system [High performance computing for multiprocessor multi-core systems]: Uchebnik [Tutorial] – M.: Izdatel'stvo Moskovskogo universiteta, 2010. – 544 p.,

4. Egunov V.A. Ocenka effektivnosti programmnoj realizacii QR-razlozheniya na mnogoyadernyh arhitekturah [Evaluating the effectiveness of software implementation of QR-decomposition on multicore architectures] / V.A. Egunov, S.I. Kirnosenko, P.D. Kravchenya, O.O. Shumejko // Izvestiya VolgGTU. Ser. Aktual'nye problemy upravleniya, vychislitel'noj tekhniki i informatiki v tekhnicheskih sistemah.[In Collection: Actual problems of control, computer engineering and Informatics in technical systems] - Volgograd, 2017. - № 1 (196). - pp. 56-59.

5. Ivanov A.M., Hohlov N.I. Primenenie tekhnologij parallel'nogo programmirovanii dlya sistem s obshchej pamyat'yu pri reshenii giperbolicheskih sistem uravnenij [Application of parallel programming technologies for systems with shared memory in solving hyperbolic systems of equations]. / Trudy Moskovskogo fiziko-tekhnicheskogo institute [Inc collection of Moscow Institute of physics and technology]. 2016. V.8 №2 (30). pp.101-111.

6. Il'in V.P. Problemy vysokoproizvoditel'nyh tekhnologij reshenij bol'shih razrezhennyh SLAU [Problems of high-performance technology solutions of large sparse systems of linear algebraic equations]. Vychislitel'nye metody i programmirovanie: novye vychislitel'nye tekhnologii [Computational methods and programming: new computational technologies]. 2009. V.10 №1. pp.141-147.

7. Kochura A. E., Podkol'zina L. V., Ivakin Ya. A., Nidziev I. I. Razrabotka algoritma resheniya sistem linejnyh uravnenij s var'iruemymi parametrami, ispol'zuyushchego razrezhennost' matricy //Prikaspijskij zhurnal: upravlenie i vysokie tekhnologii- 2014-№2

8. Anderson M. Communication-avoiding QR decomposition for GPUs / Anderson, M., Ballard, G., Demmel, J., Keutzer, K.,(2011) Proceedings - 25th IEEE International Parallel and Distributed Processing Symposium, IPDPS 2011, art. no. 6012824, pp. 48-58. ISBN: 978-076954385-7

9. Baker C.G. Anasazi software for the numerical solution of large-scale eigenvalue problems / Baker, C.G., Hetmaniuk, U.L., Lehoucq, R.B., Thornquist, H.K., (2009) ACM Transactions on Mathematical Software, 36 (3), art. no. 13. doi: 10.1145/1527286.1527287

10. Blackford L.S. An Updated Set of Basic Linear Algebra Subprograms / Blackford, L.S., Demmel, J., Dongarra, J., Duff, I., Hammarling, S., Henry, G., Heroux, M., Whaley, R.C., (2002) ACM Transactions on Mathematical Software, 28 (2), pp. 135-151. doi: 10.1145/567806.567807

11. Chen C. LU factorization on heterogeneous systems: an energy-efficient approach towards high performance / Chen, C., Fang, J., Tang, T., Yang, C. , 2017, Computing, 99(8), pp. 791-811

12. Chow E. Fine-grained parallel incomplete LU factorization / Chow, E., Patel, A., (2015) SIAM Journal on Scientific Computing, 37 (2), pp. C169-C193. doi: 10.1137/140968896

13. Demmel J. Avoiding communication in sparse matrix computations / Demmel, J., Hoemmen, M., Mohiyuddin, M., Yelick, K., (2008) IPDPS Miami 2008 - Proceedings of the 22nd IEEE International Parallel and Distributed Processing Symposium, ISBN: 978-142441694-3 doi: 10.1109/IPDPS.2008.4536305

14. Egunov V.A.. Implementation of QR and LQ decompositions on shared memory parallel computing systems [Электронный ресурс] / V.A. Egunov, A.E. Andreev // 2016 2nd International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM) (Chelyabinsk, Russia, 19-20 May 2016). – [Publisher: IEEE], 2016. – 5 p. – DOI: 10.1109/ICIEAM.2016.7911607.

15. Kreutzer M. A unified sparse matrix data format for efficient general sparse matrix-vector multiplication on modern processors with wide simd units / Kreutzer, M., Hager, G., Wellein, G., Fehske, H., Bishop, A.R., (2014) SIAM Journal on Scientific Computing, 36 (5), pp. C401-C423. doi: 10.1137/130930352

16. Kreutzer M. Building Blocks for High Performance Sparse Linear Algebra on Heterogeneous Systems / Kreutzer, M., Thies, J., Röhrig-Zöllner, M., Shahzad, F. , Pieper, A. Galgon, M., Basermann, A., Fehske, H , Shahzad, F., 2017, International Journal of Parallel Programming 45(5), pp. 1046-1072

17. Kreutzer M. Performance Engineering of the Kernel Polynomal Method on Large-Scale CPU-GPU Systems / Kreutzer, M., Pieper, A., Hager, G., Wellein, G., Alvermann, A., Fehske, H. (2015) Proceedings - 2015 IEEE 29th International Parallel and Distributed Processing Symposium, IPDPS 2015, art. no. 7161530, pp. 417-426. ISBN: 978-147998648-4 doi: 10.1109/IPDPS.2015.76

18. Luciani X. Joint Eigenvalue Decomposition of Non-Defective Matrices Based on the LU Factorization With Application to ICA / Luciani, X., Albera, L., (2015) IEEE Transactions on Signal Processing, 63 (17), art. no. 7118226, pp. 4594-4608. doi: 10.1109/TSP.2015.2440219

19. Monakov A. Automatically tuning sparse matrix-vector multiplication for GPU architectures / Monakov, A., Lokhmotov, A., Avetisyan, A. , (2010) Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 5952 LNCS, pp. 111-125. ISBN: 3642115144; 978-364211514-1 doi: 10.1007/978-3-642-11515-8_10

20. Nelson T. Reliable generation of high-performance matrix algebra / Nelson, T., Belter, G., Siek, J.G., Jessup, E., Norris, B. (2015) ACM Transactions on Mathematical Software, 41 (3), art. no. 18. doi: 10.1145/2629698

21. Polizzi E. Density-matrix-based algorithm for solving eigenvalue problems / (2009) Physical Review B - Condensed Matter and Materials Physics, 79 (11), art. no. 115112. doi: 10.1103/PhysRevB.79.115112.